REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(LEAVE BLANK)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	4 June 1999	Professional Paper

4. TITLE AND SUBTITLE

Low Cost Solutions for Automation of Simulation Test and Reporting

5. FUNDING NUMBERS

6. AUTHOR(S)

Richard Brandon Munday

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Air Warfare Center Aircraft Division
22347 Cedar Point Road, Unit #6
Patuxent River, Maryland 20670-1161

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Air Systems Command
47123 Buse Road, Unit IPT
Patuxent River, Maryland 20670-1547

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

High efficiency in the conduct of flight and simulation test usually involves high costs to achieve. Traditional methods of test automation often involve expensive, customized hardware and software systems, requiring lengthy procurement and long lead times. However, automation is highly desirable as it leads to repeatability, minimizes schedule uncertainty, and maximizes asset use. Four separate areas were targeted for low-cost automation of simulation testing: Test setup, test conduct, data retrieval, and data reduction. The automation processes used were chosen based upon the following criteria: minimal program delays, minimal training, minimal workload to implement, flexibility for future uses, and compatibility with existing systems and methods. Through the use of automated processes on existing hardware and software, total time and cost required to complete complex simulation tests were greatly reduced while improving the quality of results.

14. SUBJECT TERMS

Controls Analysis and Simulation Test Loop Environment (CASTLE)
Compact Disk - Recordable (CD-R) Digital Flight Data Computer (DFDC)
File Transfer Protocol (FTP) Maneuver Generator (Mangen)
Manned Flight Simulator (MFS) Multi-Reconfigurable Cockpit (MRC)
Visual Basic (VB) Simulation Test Flight Test

15. NUMBER OF PAGES

14

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

LOW COST SOLUTIONS FOR AUTOMATION OF SIMULATION TEST AND REPORTING

Richard Brandon Munday
Aerospace Engineer
Naval Air Warfare Center, Aircraft Division, Patuxent River MD 20670
Telephone: 301-342-3553 E-mail: mundayrb@navair.navy.mil

1. ABSTRACT

High efficiency in the conduct of flight and simulation test usually involves high costs to achieve. Traditional methods of test automation often involve expensive, customized hardware and software systems, requiring lengthy procurement and long lead times. However, automation is highly desirable as it leads to repeatability, minimizes schedule uncertainty, and maximizes asset use. Four separate areas were targeted for low-cost automation of simulation testing: Test setup, test conduct, data retrieval, and data reduction. The automation processes used were chosen based upon the following criteria: minimal program delays, minimal training, minimal workload to implement, flexibility for future uses, and compatibility with existing systems and methods. Through the use of automated processes on existing hardware and software, total time and cost required to complete complex simulation tests were greatly reduced while improving the quality of results.

2. ACRONYMS AND ABBREVIATIONS

CASTLE	Controls Analysis and Simulation Test Loop Environment
CD-R	Compact Disk - Recordable
DFDC	Digital Flight Data Computer
FTP	File Transfer Protocol
Mangen	Maneuver Generator
MFS	Manned Flight Simulator
MRC	Multi-Reconfigurable Cockpit
NAWCAD	Naval Air Warfare Center, Aircraft Division
PC	Personal Computer
VB	Visual Basic

3. INTRODUCTION

High efficiency in the conduct of flight and simulation test usually involves high costs to achieve. Traditional methods of test automation often involve expensive, customized hardware and software systems, requiring lengthy procurement and long lead times. However, automation is highly desirable as it leads to repeatability, minimizes schedule uncertainty, and maximizes asset use.

This discussion will focus on experiences during development of the S-3B Digital Flight Data Computer (DFDC) and the use of a simulator during DFDC developmental testing. The discussion will be primarily relevant to both manned (pilot-in-the-loop) and unmanned (offline) simulation of a typical military jet aircraft.

4. TEST FACILITIES AND EQUIPMENT

The simulation facility used was the Naval Air Warfare Center (NAWCAD), Patuxent River, MD Manned Flight Simulator (MFS). The S-3B is a high-wing, subsonic sea control aircraft. The DFDC is designed to replace an aging and unreliable flight control computer. The S-3B airplane pilot station was simulated at MFS by a fixed base, multi-reconfigurable cockpit (MRC). The essential S-3 pilot instrument panel and center instrument panel indicators, displays, and controls were simulated using a 27-inch, touch-sensitive video display and Virtual Avionics Prototyping Software. S-3 pilot side console and center console controls were simulated with generic hardware controls. The throttles and pilot stick grip were actual S-3 airplane parts that were integrated into the MRC. Cockpit control stick and pedal forces were simulated by an MFS-designed control loading system. The visual scene was projected onto a three-piece flat panel display with a 165 deg horizontal and 40 deg vertical field of view. The simulation software was originally based on the S-3B Weapon System Trainer (Device 2F92B) and was extensively modified by NAWCAD to support DFDC control law development and testing. This baseline FORTRAN simulation was designed to replicate the primary and secondary flight control systems, engine performance, and aerodynamics of the S-3B. The entire simulation was hosted in the Controls Analysis and Simulation Test Loop Environment (CASTLE) at MFS. During various stages of the testing, the DFDC control laws were either simulated in FORTRAN software, or performed directly by flight hardware integrated into the test simulation.

5. APPROACH TO SIMULATION AUTOMATION

Early simulation testing on the DFDC was conducted in a flight test style, in some cases from takeoff to landing, and in other cases starting at altitude but requiring the pilot to manually trim to conditions for each test point. Setups were often manually created and saved individually by the simulation engineer. Maneuver generator files were built and saved individually by the test engineer while sitting at a host computer terminal. There was no direct connection between the test matrix and the setup files, other than a list passed to the simulation engineer detailing certain critical flight conditions. This style of testing was lengthy and costly, and allowed poor repeatability between test phases.

Four separate areas were targeted for low-cost automation: test setup, test conduct, data retrieval, and data reduction. Discussion will focus first on considerations for automation, and then on how these suggestions were implemented for the DFDC program. Many of the techniques described were developed and implemented incrementally over the approximately four years of testing.

5.1 Test Setup

Setup for most flight-style testing is typically a lengthy process requiring both ensuring the simulation is running properly, and ensuring accurate flight conditions are simulated. The setup method is naturally based on the particular simulation in use, but most simulation setup routines include a combination of manual settings changes both in the cockpit and at a computer workstation, and execution of automated setup routines on the host computer. Several key areas which must be set include cockpit switches, pilot flight controls, visual environment, and flight conditions.

5.1.1 Cockpit Switches

Cockpit switches are usually manually reset due to their mechanical nature; not much can be done to automate this task. However, in some cases, as in the S-3B simulation at MFS, many of the cockpit switches were simulated on a touch-sensitive computer screen and could be set electronically. This capability greatly improved the ability to automate test setup. There is an obvious tradeoff between this setup capability and the reduced simulation fidelity offered by touch-screen technology. Although mechanical switches are more representative of flight hardware, some software simulation of switches may be an inexpensive and useful way to improve testing speed for programs where assessing the flying qualities is more important than assessing the pilot workload.

5.1.2 Pilot Flight Controls

Control stick position, pedal position and throttle lever position can vary for different trimmed flight conditions; this mandates some position correction prior to starting the simulation. Since most flight simulators currently in use offer back-driven controls for the pilot stick, throttle, and pedals, setting the flight control positions is a normal part of setup routines. The ability to back-drive all key flight controls is essential to test automation where a cockpit is used.

5.1.3 Visual Environment for Piloted Tests

Depending on test requirements, the visual scene may need to be reset for different tests. This is inherently a set of software changes, and the ability to automate this process is highly dependent on the simulation design. Most simulations will automatically set the visual scene to match aircraft attitude and position, but certain elements such as active runway or runway marker lighting need to be customized for various tests. Allowing automation of the visual environment setup requires coordination with the simulation designers to ensure that some form of macro control over key visual environment variables is available at run time. For offline testing, the visual scene is irrelevant, thus decreasing the setup requirements and reducing the complexity of any automation.

5.1.4 Flight Conditions

At the heart of simulation setup is setting the flight conditions for each test point. Simulation testing can be approached from two distinctly different directions. It can be treated as an extension of flight test, where the pilot treats the simulator as just another aircraft, and starts from a fixed runway position and flies a mission to completion. On the other hand, simulation can be treated as a rapid-fire data gathering exercise where the pilot takes control at each trimmed flight condition, executes the test technique, and then waits for the next setup. Both approaches have validity for any test program; the flight-style approach is excellent for qualitative testing and assessing overall handling qualities, while the spot-testing technique is excellent for rapidly gathering quantitative data at key flight conditions. Automation does not buy much time savings for a flight-style technique, but offers huge time savings for the spot-testing technique, where a change in setup occurs frequently, by requiring less pilot time to trim to new flight conditions. In fact, having a rapid and efficient setup allows greater use of spot-testing. This speeds the collection of numerical data, leaving more time for a thorough qualitative assessment of the system under test.

5.2 Test Conduct

The automation of test conduct depends largely on the ability to automate the setup process. However, once the simulation setup capability has been established, there are several areas where test conduct can also be automated. All these steps require coordination with the simulation designer, or some ability to modify the simulation.

5.2.1 Starting The Simulation

For offline testing, the setup macro can include a "run" command which starts the simulation at the selected flight conditions. For piloted testing, the pilot needs time to scan his instruments and controls, and prepare to start the maneuver. For this reason, simply allowing the pilot direct control over the simulation may improve test efficiency.

5.2.2 Executing the Maneuver

For piloted testing the pilot is responsible for executing the desired maneuver. For offline tests, the computer must conduct the maneuver, and understanding the simulation's maneuver generation capabilities is essential to test automation.

5.2.3 Collecting Data

Whenever digital data from a simulation is available, there are some possibilities to speed its collection and storage. First, pilot control over the data storage can reduce the number of junk data files, by allowing him to start and stop data collection as appropriate. Second, the storage of collected data can be automated. Eliminating the requirement to manually name and save each file will reduce the delay between test points. This requires establishing a logical data file naming scheme and automatic filename generation to ensure that data is not lost or misplaced.

5.2.4 Stopping and Resetting the Simulation

A useful feature for test automation is a predefined run length. Designing the simulation to stop automatically and await further commands allows the test designer to write a macro that simply calls the simulation to execute and waits for test point completion before issuing the next command. Once the maneuver is complete and the data has been stored, the simulation must be reset for the next run. Caution must be taken while resetting the simulation to reset all variables which may have been directly or indirectly changed during the test run.

5.2.5 Repeating the Test

Often during piloted testing, the maneuver must be repeated at the same flight conditions. Giving the pilot control over the reset function allows him to make several attempts at the maneuver until the desired data has been collected. In particular, for difficult test techniques, this allows the pilot to try varying inputs at repeatable conditions until he finds the desired input. This capability is much more efficient than requiring the pilot to manually retrim to starting conditions after each attempt. A potential pitfall here is negative training; an overly easy reset capability masks the difficulty of obtaining similar data during flight tests. The simulation may become a poor predictor of overall flight test conduct, leading to an overly aggressive flight test schedule.

5.3 Data Retrieval

The data retrieval process typically involves downloading digital data files from a simulation host computer to an engineer's desktop PC. Considerations for speeding this process include file naming conventions and network access to the host computer. Simulation host computers typically use UNIX or SUN operating systems, which use different file naming conventions from the average PC. Some conversion of file names or extensions may be required when the files are transferred. Also, a networked connection to the host computer can speed the transfer of data, compared to transfer using some form of removable tape or diskette.

5.4 Data Reduction

Data reduction is the key to any test program. Until recently, the ubiquitous strip chart was the primary method of data collection for real-time testing. The strip chart, however, has declined in importance with the advent of large-scale collection and storage of digital data. Also, with the capability to conduct offline simulation testing faster than real-time, digital data storage is essential. Strip charts are costly in several areas: they require full-time monitoring and marking during testing, they require excessive space to store, correlating data across numerous charts is tedious, and converting strip chart data to report-quality data is slow and prone to mistakes. However, digital data has its own problems: rapid visualization of data is difficult, and storage is physically compact but costly.

Although there are numerous areas where data reduction can be streamlined, one key area is the initial overview of the collected data. For efficient assessment of the results of a simulation test, some portion of the data is usually plotted, typically as a time history graph. Automating the plotting process can provide significant time savings. An automated plotting routine can allow all data files collected on a given test to be bulk-plotted with little user interaction, and frees the engineer for other tasks. Workload can be further reduced by tailoring each plot to the test type, rather than generating a generic set of plots for all tests. Also, generating report-quality plots on the first pass will minimize the work required if a plot is later chosen for inclusion in a report. Finally, if the tools used produce high-quality graphical output, those tools can later be used directly in the reporting process.

6. SOLUTIONS IMPLEMENTED

The automation processes used in the DFDC program were chosen based upon the following criteria: minimal program delays, minimal training, minimal workload to implement, flexibility for future uses, and compatibility with existing systems and methods. The NAWCAD MFS simulation used for DFDC testing was ripe for easy implementation of many automation techniques. Other automation techniques were developed for the engineer's data collection and reporting processes as well.

6.1 Selection of Software Tools

Since most onsite desktop computers had Microsoft Office installed, the Office suite was instrumental in building the custom solutions required. Visual Basic (VB) for Microsoft Office was chosen to create various macros. It provided a simple, highly flexible, and easily learned tool, provided cross-platform capabilities not tied to a specific computer type, used common and familiar interface elements, allowed the creation of customized numerical processing and graphical output, and was powerful enough to allow speedy completion of various tasks.

6.2 Test Setup Methods Used

6.2.1 File Formats

The first area of automation undertaken for the DFDC program was determining setup file formats for the simulation host computer. Two key types of setup files used by the CASTLE host computer were stored in easily interpreted, unformatted text files: "command files" and maneuver generator or "mangen" files. The command files functioned as if the simulation engineer were typing the included commands manually. Mangen files required more interpretation but were also fairly simple in structure. Once the format and structure of these files were understood, they could be generated offline and uploaded to the host.

6.2.2 File Generation Tools Created

Next, tools were developed to create the command and mangen files. One key to this process was a consistent and standardized test matrix format. Using VB macros, the test matrix was converted from a Word document into an Excel spreadsheet of test conditions, which contained one row for every planned test point. Information was then added to the table for each test point, including run length, data file names, desired location relative to airfields or carriers, mangen filenames, and so forth. Then a VB macro was written that saved a command file for each test point, directly in the CASTLE file format. Two types of command files were generated: one for piloted tests, and another for offline tests. These command files included commands to load generic setup files, change flight conditions, change various cockpit controls, and trim to the new flight conditions. Additional commands were added to the offline files, including commands to load mangen files, set run length, start the simulation, change host disk directories, and store data files on the host disk with preset names. Multiple offline tests could be conducted sequentially by one command file, allowing the simulation engineer to start the command file and carry out other tasks while the host computer managed many hours of testing.

Once the file generation tools were complete, it became possible to update or rebuild any setup files simply by changing the table of test conditions, and re-running the command file generation macro. In particular, it was simple to precisely duplicate previous offline test events by reusing the appropriate maneuver generator files with the latest test matrix. Regression testing was precise enough that most offline test data could be plotted directly over prior test results with perfect matches in most cases.

6.2.3 Test Preparation

Once the capability to automate the setup and test conduct had been established, the initial test preparation workload for the test engineer increased. File naming schemes and variable names were chosen. Maneuver generator and setup files had to be built and double-checked, and then transferred to the simulation host computer. Although the setup time for the first evaluation that used these techniques was lengthy, there was virtually no setup time required (less than one man-day) for subsequent test cycles. Once the test team had finalized the test matrix, a day or two was enough to produce the new set of command files for the new test sequence.

6.2.4 Sample File Structures

For offline testing, the mangen files defined the variables to be changed, the input shape, and the time sequence for the input. A VB graphical user interface was developed in Excel to create

and modify the mangen files. This allowed the engineer to work at his desk using a familiar mouse-driven interface, rather than the clumsy command line text-based method required by CASTLE. A typical mangen file is shown in figure 1.

Figure 1
Maneuver Generator File Format

*	Module	Parameter	Function	S.Time	E.Time	Start_Amp	End_Amp	Bias	Start_Dur	End_Dur	Frq_Code
MANGEN	*	FMS_YAW_CMD	STEP	1	2	3	0	0	1	0	2
MANGEN	*	FMS_ROLL_CMD	STEP	2	3	1	0	0	1	0	2
MANGEN	*	PLA_L_CKP	RAMP	10	99	-100	0	IC	1	0	2
MANGEN	*	PLA_R_CKP	RAMP	10	99	-100	0	IC	1	0	2
MANGEN	*	FMS_PITCH_CMD	STEP	1	2	1	0	0	1	0	2
MANGEN	*	FLAT_CMD	STEP	3	6	5	0	0	3	0	2
MANGEN	*	SPDBRK_SW_AFT	STEP	10	109	-1	0	IC	99	0	2

Command files were generated by the VB macros directly from the table of flight conditions. A typical command file was formatted as shown in figure 2. It includes loading generic settings files, customizing the run length, selecting data variables to be stored and the data storage rate, customizing the flight conditions and airplane configuration, trimming the simulation, running four sequential offline tests with individual mangen files, and saving each set of data with a unique and relevant filename.

Figure 2
Command File Format

```
LOAD [USERS.BASE.PILOTED.SET_LOADING]LOADING_A.SET
SET ICC LOAD [USERS.BASE.PILOTED.ICC]OFF_HWL_DFDC.ICFG
SET ICS LOAD [USERS.BASE.PILOTED.ICS]INFLIGHT.ISS
SET STORE LOAD [USERS.BASE.PILOTED.SSF]EVAL_ALL.SSF

SET STORE INC 12

SET PARM CHANGE TRUN 60
SET PARM CHANGE DO_STORE .TRUE.
SET PARM CHANGE DT 0.00416666667

SET ICST CHANGE VCAL 225
SET ICST CHANGE ALT 15000
SET ICST CHANGE PHI 0

SET ICA CHANGE MFLAP 1
SET ICA CHANGE DLC_ENGAGE_CMD .FALSE.
SET ICA CHANGE GEAR_HAND_DWN .FALSE.
SET ICA CHANGE HOOK_HAND_DWN .FALSE.
SET ICA CHANGE USGW 40000
SET ICA CHANGE USCG 23

TRIM

SET MAN LOAD [USERS.BASE.OFFLINE.DFDC_V21.MSF]DFDCV2_1202A1.MSF
RUN
STORE/ASCII [USERS.BASE.OFFLINE.DFDC_V21.DATA]DFDCV21_1202A_01.TXT

RESET
SET MAN LOAD [USERS.BASE.OFFLINE.DFDC_V21.MSF]DFDCV2_1202A2.MSF
RUN
STORE/ASCII [USERS.BASE.OFFLINE.DFDC_V21.DATA]DFDCV21_1202A_02.TXT

RESET
SET MAN LOAD [USERS.BASE.OFFLINE.DFDC_V21.MSF]DFDCV2_1202A3.MSF
RUN
STORE/ASCII [USERS.BASE.OFFLINE.DFDC_V21.DATA]DFDCV21_1202A_03.TXT

RESET
SET MAN LOAD [USERS.BASE.OFFLINE.DFDC_V21.MSF]DFDCV2_1202A4.MSF
RUN
STORE/ASCII [USERS.BASE.OFFLINE.DFDC_V21.DATA]DFDCV21_1202A_04.TXT
```

A "meta-command" file was developed as well which allowed individual command files to be run as a group. Similar meta-files allowed an entire section of test methods to be run. This allowed the simulation engineer to start an overnight batch processing job and essentially walk away for up to eight hours while testing was conducted automatically. This technique allowed efficient use of busy test assets at times when other MFS testing would not be impacted.

6.3 Test Conduct Methods Used

Command files were used for rapid setup and trim at each new flight condition, unless the pilot wanted to fly between points for qualitative evaluation. For offline tests, the command file also loaded a mangen file prior to initiating the simulation run; CASTLE automatically invoked the maneuver generator at the defined time. Extra stick grip switches were used to allow the pilot to start, stop and reset the simulation, and to start and stop data storage. However, there was no automatic file saving, requiring the simulation engineer to manually name and save each file during piloted testing. The CASTLE "reset" command for the S-3B simulation only reset the flight condition and aircraft control states, and cockpit switches and certain other controls had to be reset manually for piloted testing. The process of trimming the flight condition automatically initiated a reposition of the flight control positions to match computed positions. However, early in the S-3B program the throttle levers were not automatically repositioned; this caused increased test time due to the requirement for the pilot to retrim to the desired flight conditions after each simulation reset. Addition of back-driven throttle levers greatly improved efficiency while resetting during piloted tests. For offline testing, cockpit control and switch positions were ignored; this allowed software changes in control and switch position without concern for resetting the cockpit after the maneuver.

6.4 Data Retrieval Methods Used

File Transfer Protocol (FTP) was used to download data files from the CASTLE host computer and to upload settings files. Zip disks and CD-R disks were used to transfer data between test and simulation engineers. File naming differences between the host computer and the engineer's PC were handled by careful selection of an FTP program which could be configured to translate filename extensions. Additionally, an Excel macro was developed to further rename the files as desired in a batch fashion.

6.5 Data Reduction Methods Used

6.5.1 Selection of Software

Several data plotting packages were surveyed as potential candidates for conducting the data analysis, including MATLAB, KaleideGraph, DeltaGraph, and Excel. Excel was chosen for the following reasons.

Efficient: Excel had built-in wizards that allowed simple import and graphing of the text-based data files. MATLAB was more efficient at handling large text files but difficulty scripting and customizing its output was a significant deterrent. Most notably, the Excel output could be added to and saved with the raw data file, eliminating the need to rebuild the plots for later data reduction.

User-Friendly: Using VB's dialog boxes, a graphical user interface could be constructed to allow simple and logical control of the customized scripts. VB is a fairly simple language to learn and Excel's macro recording capability allows the user to learn VB by example.

Scriptable: Every function Excel performs could be scripted using VB, allowing efficient and customized repetition of tasks. MATLAB offered scripting capabilities, but it used a cryptic command-line-based user interface.

Customizable: The results of any Excel operation could be customized and the graphical output could be made to exactly suit the test engineers' needs. The other packages produced graphs that were impossible to customize to NAWCAD standards, or required excessive work to customize.

Cost-Effective: Excel was already installed on engineers' machines as part of a Microsoft Office site license, requiring no additional spending to procure.

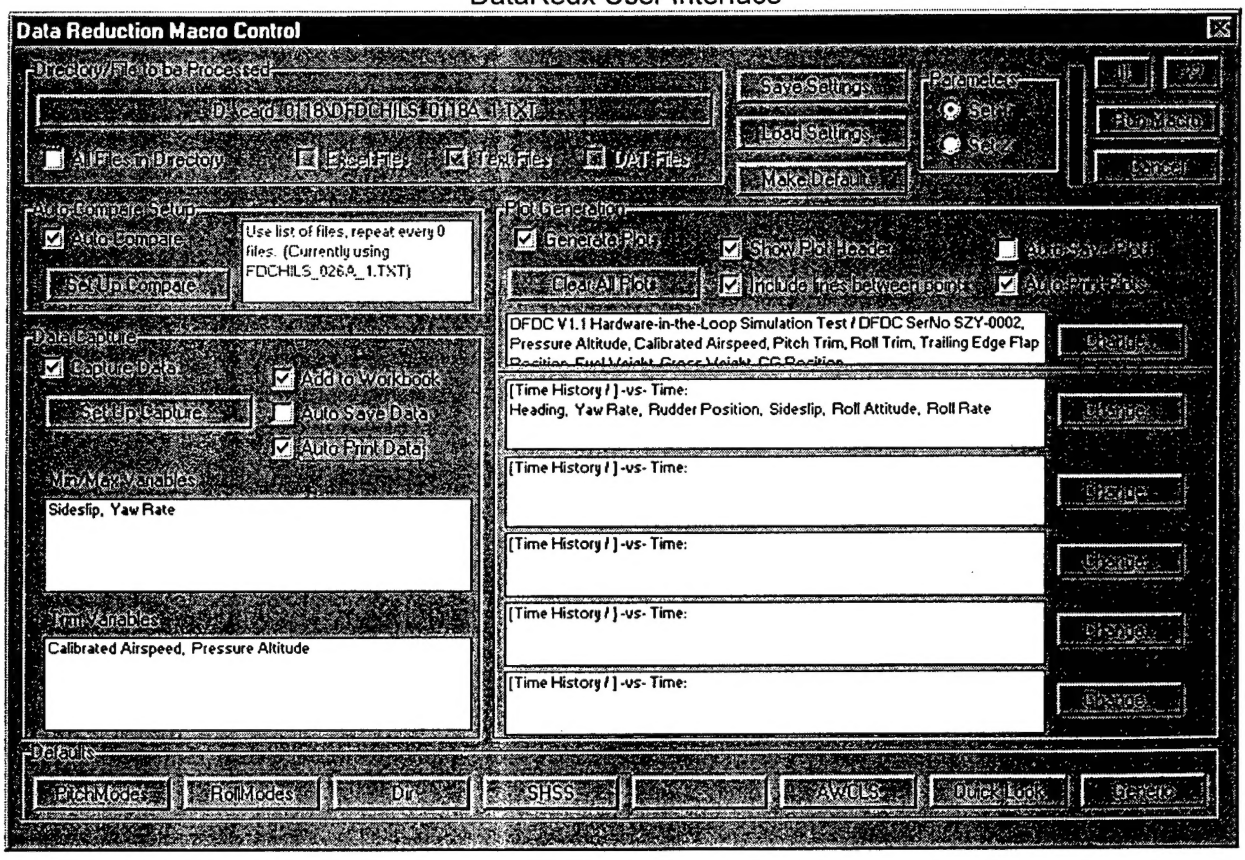
6.5.2 Creation of Macros

An Excel VB macro, internally called DataRedx, was developed for indexing, loading, plotting, and overlay-plotting entire directories full of digital data files. It offered a comprehensive ability to select desired parameters, build up to six plots per page and five total pages per data file, produce report-ready graphs, and save and load all settings. In addition, to help automate some data reduction requirements, it could capture local and absolute minimum and maximum values for selected parameters. All the results could be automatically printed or saved in the data file itself. It was also completely customizable to keep up with changes in test requirements and the CASTLE environment.

6.5.3 Macro Tools Developed

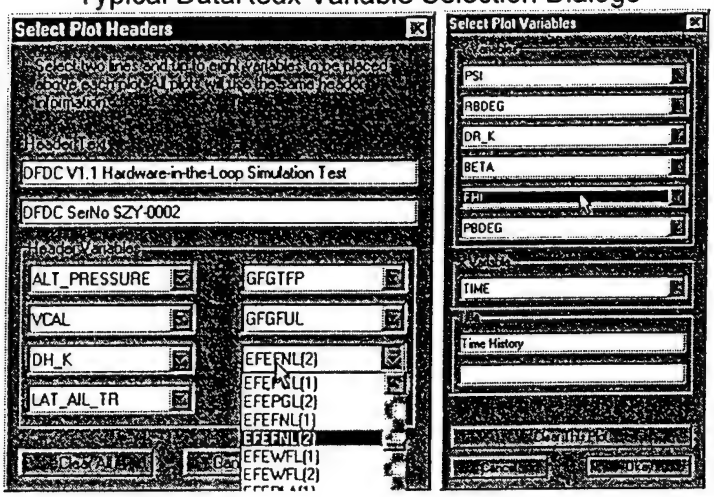
A view of the user interface for the DataRedx macro is shown in Figure 3.

Figure 3
DataRedx User Interface



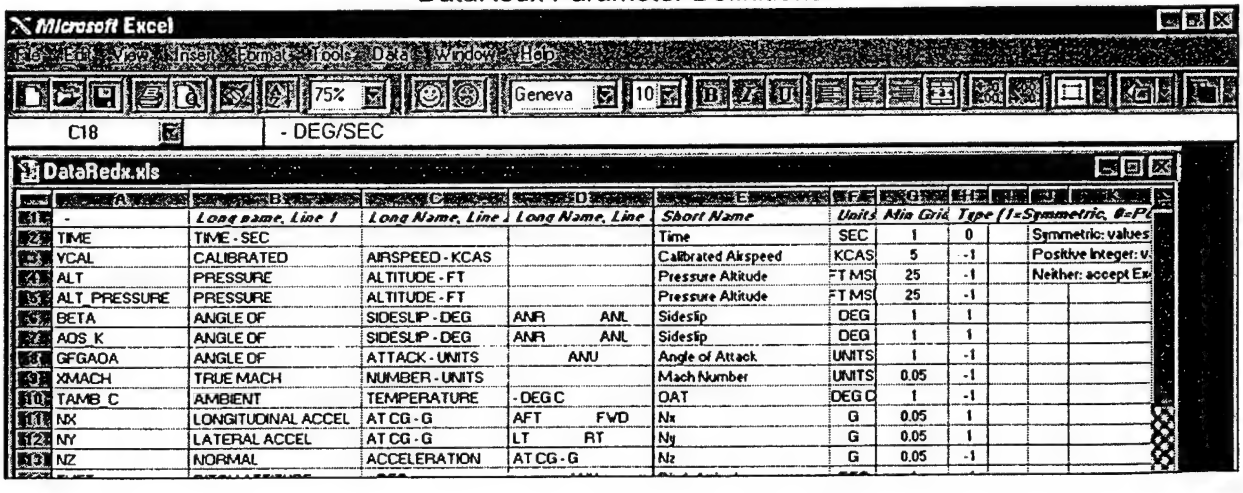
Two typical variable selection dialogs are shown in figure 4. Dropdown lists were provided for simple selection of up to eight "state" variables in the header, and provision was made for a customized two-line header and title for each plot. Provision was also made for cross-plots against variables other than Time, and if less than six variables were requested the plots would be automatically scaled to fill the page.

Figure 4
Typical DataRedx Variable Selection Dialogs



A list of parameters was used to customize DataRedx to the specific tests being conducted; a sample is shown in figure 5. The left column defined the variable mnemonic stored in the data file. The middle columns defined the names and variable definitions which appeared on printed output. Extra columns on the right defined how the plots were generated for that parameter. Parameters added to this list automatically appeared on the dropdown lists in the plot selection and plot header dialogs. Once this list was completed, each data file was parsed to find the correct columns to plot, making the macro results independent of data file format. The same plot routine was later used for flight test data files with little customization other than updating the parameter list.

Figure 5
DataRedx Parameter Definitions



	A	B	C	D	E	F	G	H	I	J	K
1	-	Long name, Line 1	Long Name, Line 2	Long Name, Line 3	Short Name	Units	Min Grid	Type (1=Symmetric, 0=Pl			
2	TIME	TIME - SEC			Time	SEC	1	0		Symmetric: values	
3	VCAL	CALIBRATED	AIRSPEED - KCAS		Calibrated Airspeed	KCAS	5	-1		Positive Integer: v	
4	ALT	PRESSURE	ALTITUDE - FT		Pressure Altitude	FT MSL	25	-1		Neither: accept Ex	
5	ALT_PRESSURE	PRESSURE	ALTITUDE - FT		Pressure Altitude	FT MSL	25	-1			
6	BETA	ANGLE OF	SIDESLIP - DEG	ANR ANL	Sideslip	DEG	1	1			
7	AOS_K	ANGLE OF	SIDESLIP - DEG	ANR ANL	Sideslip	DEG	1	1			
8	GFGAOA	ANGLE OF	ATTACK - UNITS	ANU	Angle of Attack	UNITS	1	-1			
9	XMACH	TRUE MACH	NUMBER - UNITS		Mach Number	UNITS	0.05	-1			
10	TAMB_C	AMBIENT	TEMPERATURE	- DEG C	OAT	DEG C	1	-1			
11	NX	LONGITUDINAL ACCEL	AT CG - G	AFT FWD	Nx	G	0.05	1			
12	NY	LATERAL ACCEL	AT CG - G	LT RT	Ny	G	0.05	1			
13	NZ	NORMAL	ACCELERATION	AT CG - G	Nz	G	0.05	-1			

An unedited sample plot as produced by the DataRedx macro is shown in figure 6. These plots were generated by the thousands with no user intervention other than adding paper to the printer. Over 15,000 pages of printed output were generated and reviewed during the course of four simulation evaluations, but due to the level of automation, more time was spent putting the paper in binders than was spent at the PC generating this quantity of output.

Figure 6
DataRedx Plotting Output

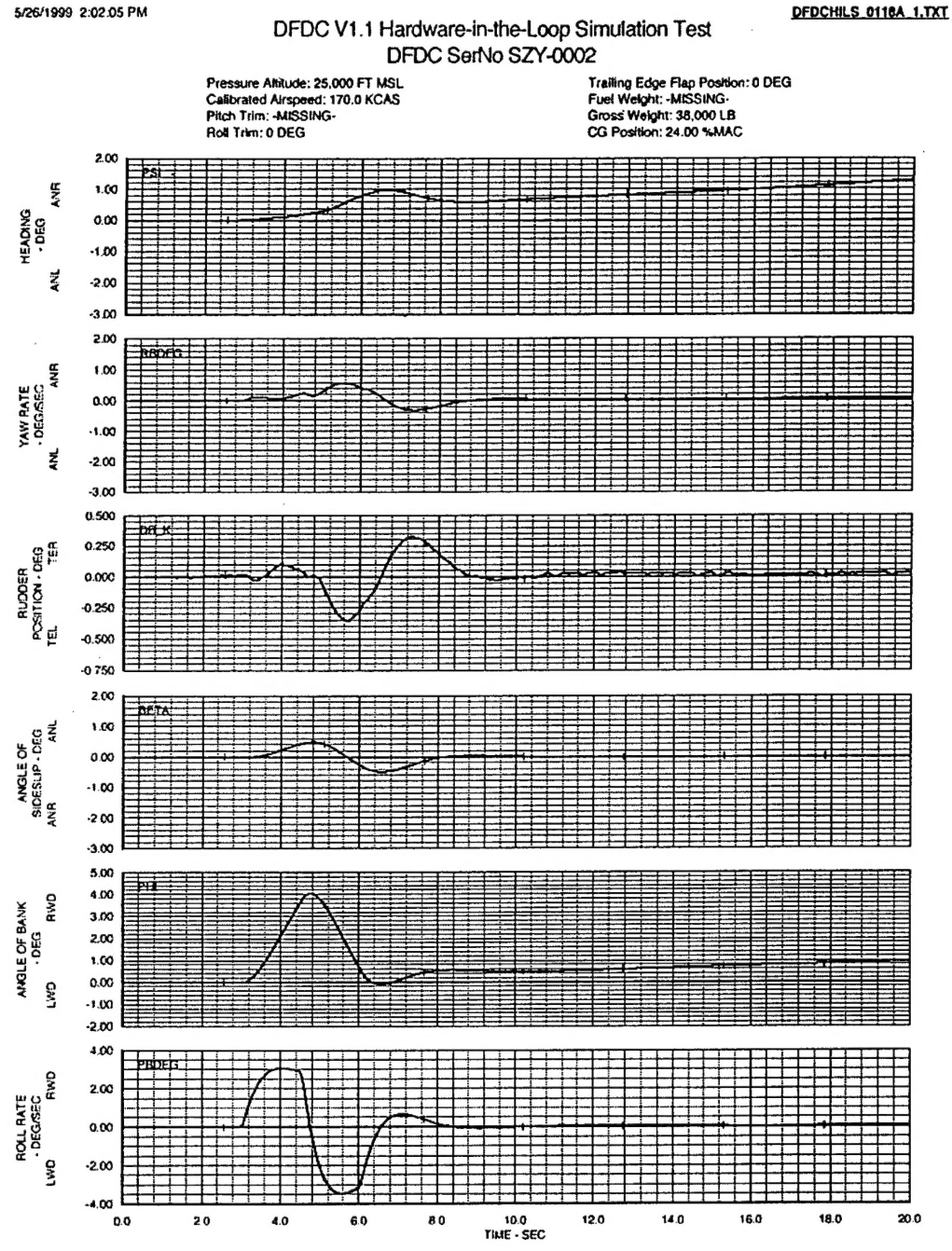


Figure 1
YDTC Dutch Roll Mode Damping, Lateral Doublets, Time History

D:\card_0118

Plot Group 1.1

7. RESULTS

Prior to implementation of the methods chosen, total time required for a typical simulation evaluation could exceed 5-10 times the actual test conduct times. 40 simulation hours to complete 4 hours of test points was not uncommon, due to the time required to set up each test, change simulated airplane configurations, fly between test points, repeat the test until adequate data quality was obtained, and save the digital data files. Using the techniques developed, total simulation time for off-line (non-piloted) points was reduced to approximately 110% of actual test time, barely 10-15% of that required during previous test phases. Much of the off-line testing was completed overnight with no supervision, thus freeing the engineers and simulation assets for other daily tasks. By making the setup process efficient, total simulation time for most piloted points was also reduced by 50-75%. Data plotting and analysis was automated to the point of generating thousands of pages of comparison plots within a few days of test completion, with minimal user interaction. Furthermore, the preliminary methods and macros developed for previous test phases will be reusable for future testing, providing further savings in budget, schedule, and workload for both the current test cycle and future projects. No special software or hardware was obtained for the project, and the results can be duplicated on almost any typical desktop computer using preinstalled software.

8. LESSONS LEARNED

- The earlier automation is applied, the greater the payoff during later testing.
- Automation can be best applied if the test engineer has input during the simulation design.
- Simple text-based settings files are important for automation of simulation testing.
- The ability to back-drive all key flight controls is essential to piloted test automation.
- Many useful features of the most common software packages remain unused by the average engineer but have the potential to make testing much more efficient.
- Automation of offline testing results in highly repeatable and easily reduced regression data.
- Efficient quantitative offline and piloted tests speed the collection of numerical data, leaving more time for a thorough qualitative assessment of the system under test.

9. FUTURE APPLICATIONS

Any simulation project that depends on repetitive setup of test conditions, lengthy test cycles, repeated regression testing, or high-volume output of graphical data, can benefit from automation, and the required methods and skills can be readily learned and employed by the average engineer. This type of automation is possible with many commonly used data collection and reduction tools, desktop software suites, and data processing programs.

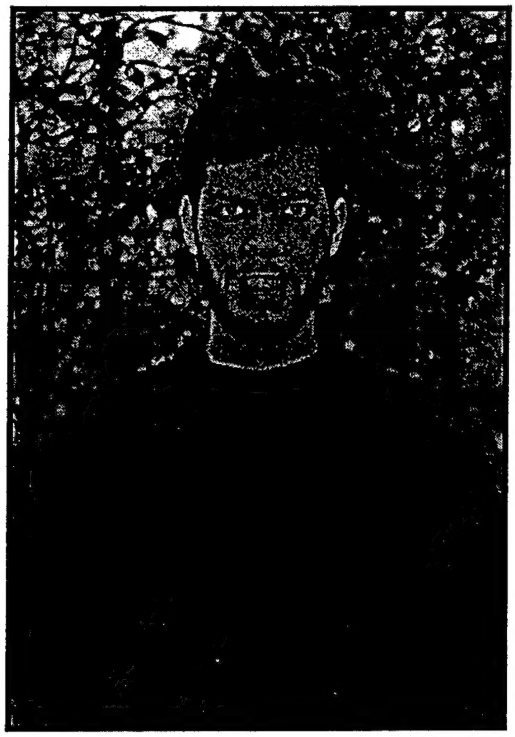
10. ACKNOWLEDGEMENTS

The following individuals participated in the development of the S-3 simulation at Manned Flight Simulator, and in creating the flexibility that allowed the test automation: Danny Campbell, Peter Gendron, Keith Balderson, Dean Liu, Jeff Weathers, Dave Luna, Terrie Paletar, and most of all, Jim Kelly.

11. CONCLUSIONS

Through the use of automated processes on existing desktop hardware and software, total time and cost required to complete complex simulation tests can be greatly reduced while the quality of results is improved. Careful evaluation of existing processes may reveal the potential for significant savings. However, a substantial initial investment of time is required to implement these savings, and should be planned into the program. Wherever possible, the test engineers should be directly involved in creation and programming of the simulation environment to ensure that test automation is possible. As aviation testing continues to shift resources away from aircraft testing and towards flight simulation, automation of simulation testing can provide real savings in schedule and costs.

BIOGRAPHY



R. Brandon Munday
Aerospace Engineer
Naval Air Warfare Center, Aircraft Division
Patuxent River, MD 20670

BS Aerospace Engineering
Virginia Polytechnic Institute and State University
Class of 1990

Mr. Munday has been employed by NAWCAD Patuxent River since his graduation from VPI&SU in 1990, and has been involved in simulation and flight testing on the T-45A, AV-8B, A–4J, and most recently the S-3B. He has authored and participated in publication of numerous technical reports on tests of those model simulators and aircraft.

Mr. Munday has been involved with computer programming since long before in his aerospace training, on languages ranging from Commodore 64 Basic, Pascal, FORTRAN and PERL to most recently Visual Basic. He considers simulation testing to be the perfect blend of his interests in computers and aircraft.

He also maintains a home business based on sales of music-related software he has written. Recently he released a Windows 95/98 music transposition utility and is preparing to release a new Windows version of a song database application.